OPENRAIL INCUBATION REVIEW

TP-Lib — Stage 1 Review

Report date: 2026-06-30

Contents

This review was conducted with AI assistance (Kiro) by Cornelius Schumacher, OpenRail Technical Committee.

Overview

PR	**#177 — doc: add tp-lib stage 1 proposal**
Submitted	2026-06-30
Applicant	Mathias Vanden Auweele, Infrabel (via Matdata)
Repo	https://github.com/Matdata-eu/tp-lib

Project summary. TP-Lib is a Rust library for post-processing GNSS positions recorded by measurement trains onto an unambiguous location in a railway network (map matching). It uses a Hidden Markov Model with Viterbi decoding (Newson & Krumm 2009) to identify the most probable path through the network. The primary language is Rust (33K lines across the workspace). The workspace includes a core library (`tp-lib-core` on crates.io), CLI (`tp-cli`), Python bindings via PyO3 (`tp-lib` on PyPI), .NET bindings via csbindgen (`TpLib` on NuGet), and a browser-based path review webapp (`tp-webapp`). It integrates with the EU Register of Railway Infrastructure (RINF) SPARQL endpoint for topology retrieval. Apache-2.0 licensed. Originates from Infrabel, the Belgian railway infrastructure manager.

Review against Stage 1 (Onboarded) criteria.

Criteria evaluation

1. Railway sector scope ✅

Map-matching GNSS data to railway track segments is a core operational problem for every infrastructure manager that operates measurement trains. The questionnaire states: “all railway infrastructure managers do this and should better do it together.” The project addresses directed, navigable railway topology where general-purpose map-matching libraries fall short. Uses railway-domain terminology (RailTopoModel, RSM, EULYNX, railML, ERA ontology). Clearly within OpenRail’s scope.

2. Clear description ✅

Extensive README (21KB) covering features, architecture, usage, and building. The questionnaire is thorough — no “TBD” answers on technical details. Detailed feature specifications in `specs/` covering GNSS projection, train path calculation, path review webapp, train detections, .NET bindings, and RINF topology download — each with research, data models, plans, tasks, and contract tests. API documentation published at https://matdata-eu.github.io/tp-lib/. A `publiccode.yml` file is present with complete metadata.

The questionnaire source is maintained in the repository at `docs/OpenRail-onboarding.md`.

The roadmap in the questionnaire is minimal (“no further development is planned at this time” aside from two community-dependent features: odometry distribution and RSM-DX I/O). Not unusual for Stage 1, but sets expectations about the project’s current phase.

3. Designated maintainers ✅

Mathias Vanden Auweele is named as maintainer (marked “ad interim”) in both the questionnaire and `MAINTAINERS.md`, with organization listed as Infrabel. Contact: mathias@matdata.eu. Single maintainer situation is common for Stage 1.

4. Code of Conduct ✅

The questionnaire’s concluding statement confirms the project will adhere to the OpenRail Association Code of Conduct. A `CODE_OF_CONDUCT.md` is already present in the repository.

5. IP Policy compliance ✅

- **License:** Apache-2.0 — OSI-approved. `LICENSE` file present and correct.
- **Copyright holder:** Not explicitly stated. The LICENSE file uses the Apache-2.0 template text without filling in the copyright holder placeholder. The `Cargo.toml` lists the author as “Mathias Vanden Auweele” and the questionnaire names Infrabel as the sponsoring organization, but no file explicitly declares copyright ownership. This should be clarified during onboarding — typically with SPDX headers naming the copyright holder.
- **Trademarks:** None.
- **Third-party code in source tree:** Leaflet.js (BSD-2-Clause) bundled in `tp-webapp/static/leaflet/`. Compatible with Apache-2.0.
- **Dependencies:** `deny.toml` enforces an allow-list of only permissive licenses: Apache-2.0, MIT, ISC, BSD-2-Clause, BSD-3-Clause, BSL-1.0, CC0-1.0, Unlicense, Zlib, Unicode-DFS-2016, Unicode-3.0, CDLA-Permissive-2.0. No copyleft dependencies.
- **CI enforcement:** The `ci.yml` workflow runs `cargo deny check` on every PR, ensuring no new license violations can be introduced.

Repository inspection

Project vitals

- **Commits:** 176, spanning 2025-12-09 to 2026-05-25 (~6 months)
- **Contributors:** 1 human (Mathias Vanden Auweele, 167 commits) + GitHub Copilot SWE agent (9 commits)
- **Stars/Forks:** 1 / 0
- **Issues:** 0
- **PRs:** 9 merged (6 by maintainer, 3 by Copilot SWE agent)

Single-developer project with active AI-assisted development. The 6-month history shows intensive feature delivery through numbered specs (001-006). No external contributors or community engagement yet.

Security findings

No secrets detected. Gitleaks scan of 163 commits (94 MB) found no leaks. Manual grep found only a `CancellationToken` code pattern (not a secret).

Hardcoded endpoint: The default RINF SPARQL endpoint (`https://graph.data.era.europa.eu/repositories/rinf-plus`) in `tp-core/src/models/retrieval.rs` is a public EU data portal — not a concern. It's configurable via CLI flag `--rinf-endpoint`.

GitHub Actions vulnerabilities (from Grype):

Action	Version	Vulnerability	Severity	Fixed in
`actions/download-artifact`	v4	GHSA-cxww-7g56-2vh6	High	4.1.3
`pypa/gh-action-pypi-publish`	release/v1	GHSA-vxmw-7h4f-hqxh	Low	1.13.0

Recommendation: pin actions to fixed versions.

Code and structure

- **Files:** 448 total (33,124 lines of Rust across the workspace)
- **Workspace layout:**
  - `tp-core/` — core algorithms (projection, path calculation, HMM/Viterbi, I/O, models); published as `tp-lib-core` on crates.io
  - `tp-cli/` — command-line interface (57K single file `main.rs`)
  - `tp-py/` — Python bindings (PyO3/Maturin); published as `tp-lib` on PyPI
  - `tp-net/` — .NET bindings (csbindgen); published as `TpLib` on NuGet
  - `tp-webapp/` — browser-based path review (axum + Leaflet, static assets compiled in)
  - `specs/` — 6 detailed feature specifications
  - `test-data/` — integration test fixtures (GeoJSON, CSV, PNG visualizations)
- **Testing:** 598 test annotations (`#[test]` + `#[tokio::test]`), with unit, integration, contract, and CLI tests. Benchmarks via Criterion in `tp-core/benches/`.
- **Docker:** `Dockerfile` and `docker-compose.yml` present for containerized deployment.
- **CI:** 7 workflows: CI (test + clippy + cargo-deny), coverage (Codecov), docs (GitHub Pages), SBOM generation, and publishing (PyPI, crates.io, NuGet).

The `tp-cli/src/main.rs` at 57K lines is large for a single file — this is a style choice but may benefit from modularization as the project grows.

AI-assisted development

The project makes extensive use of AI-assisted development tooling:

- **`.github/agents/copilot-instructions.md`:** Auto-generated from feature plans. Defines coding standards (rustfmt, clippy zero-warnings, full test suite), project structure, and active technologies. Updated with each feature spec.
- **`.specify/`:** SpecKit tooling with templates, scripts, and a "constitution" document defining the development process. Features are specified through numbered specs with research, data models, plans, tasks, contracts, and checklists.
- **`.github/agents/`:** 10 SpecKit agent definition files covering analysis, specification, planning, implementation, and task management.
- **`.github/prompts/`:** 9 prompt files for SpecKit workflows.
- **Copilot SWE agent commits:** 9 commits and 3 merged PRs from `copilot-swe-agent[bot]` (fixing docs, security/correctness issues).
- **Co-authored commits:** Multiple commits with `Co-authored-by: Copilot` trailers.

This is a well-structured AI-assisted development workflow. The specs provide clear guardrails and requirements that constrain AI-generated code. Relevant context for a single-developer project: the AI tooling provides a form of systematic development discipline.

REUSE compliance

Not compliant. 0 of 444 files have license information per REUSE spec 3.3. Only one file (`tp-webapp/static/app.js`) has a copyright notice (for bundled Leaflet code). This is an onboarding activity for Stage 2.

Secrets scan

No leaks found. Gitleaks scanned 163 commits (94 MB). Clean.

Vulnerability scan

Two GitHub Actions vulnerabilities found (see Security findings above). No vulnerabilities in Rust dependencies (checked via `cargo deny check` in CI against RustSec advisory database).

License enumeration

The SBOM enrichment step failed (GitHub API 404 for `rust-lang/crates-io-auth-action`), so automated license extraction was not possible. However, license governance is well-handled: - `deny.toml` defines a strict allow-list of permissive licenses only - `cargo deny check` runs in CI on every PR - Questionnaire explicitly lists dependency licenses (MIT, Apache-2.0, BSD-2-Clause, CDLA-Permissive-2.0) - No copyleft dependencies

Interesting dependencies

- **ureq** (HTTP client for RINF SPARQL) — blocking HTTP, appropriate for a library that doesn't need async for this use case
- **petgraph** (graph algorithms) — well-established, used for the Viterbi path through rail network topology
- **axum + tokio** (web server) — for the path review webapp; compiled in with static assets
- **proj4rs** (coordinate transforms) — Rust port of Proj, for CRS-aware geospatial calculations
- **rust-embed** — bundles Leaflet and webapp assets into the binary at compile time

Overall assessment

Positive. TP-Lib clearly meets all Stage 1 criteria. It solves a well-defined railway infrastructure problem, is comprehensively documented and specified, properly licensed with enforced dependency governance, and demonstrates active development. The single-developer situation is the main area to watch, but the project is explicitly seeking collaboration through OpenRail to address this.

Recommendations for onboarding

1. Update `actions/download-artifact` to v4.1.3+ and pin `pypa/gh-action-pypi-publish` to a fixed version to resolve the two known GitHub Actions vulnerabilities.
2. Begin REUSE compliance work (adding SPDX headers) as a post-onboarding activity toward Stage 2.
3. Add a Dependabot or Renovate configuration for automated dependency updates (Scorecard: Dependency-Update-Tool = 0).
4. Consider pinning GitHub Actions by SHA rather than tag for supply-chain security (Scorecard: Pinned-Dependencies = 0).
5. Enable branch protection rules on the main branch (Scorecard: Branch-Protection = 0).
6. The `tp-cli/src/main.rs` (57K lines) may benefit from being split into modules as the project grows.

Open questions for the applicant

1. The maintainer is listed as “ad interim” — is there a plan at Infrabel to designate a permanent maintainer, or is this expected to remain with the current person?
2. Is TP-Lib currently used in production at Infrabel for operational measurement train data processing?
3. Who is the copyright holder of the project — Infrabel, Matdata, or the individual author? This should be stated explicitly (e.g. via SPDX headers) before or during onboarding.

Appendix A: Community health files

File	Status	Notes
LICENSE	✅	Apache-2.0, standard text
README.md	✅	Comprehensive (21KB)
CODE_OF_CONDUCT.md	✅	Present
CONTRIBUTING.md	✅	Detailed (11KB)
MAINTAINERS.md	✅	Single maintainer, MVG-based
GOVERNANCE.md	✅	MVG-based consensus model
SECURITY.md	✅	Present (not required for Stage 1)

Appendix B: Dependency licenses

Automated license enumeration was not available (SBOM enrichment failed). License governance from `deny.toml`:

Allowed licenses: Apache-2.0, Apache-2.0 WITH LLVM-exception, MIT, MIT-0, ISC, BSD-2-Clause, BSD-3-Clause, BSL-1.0, CC0-1.0, Unlicense, Zlib, Unicode-DFS-2016, Unicode-3.0, CDLA-Permissive-2.0

Enforcement: `cargo deny check` in CI. Any dependency with a license not on this list will fail the build.

Per questionnaire: geo-rs, rstar, geojson, petgraph, serde, chrono, thiserror, csv (MIT OR Apache-2.0); polars, proj4rs, tracing, PyO3 (MIT); Apache Arrow (Apache-2.0); Leaflet.js (BSD-2-Clause); webpki-root (CDLA-Permissive-2.0).

Appendix C: Automated scans

Tool	Version	Status	Notes
reuse	6.2.0	❌ Not compliant	0/444 files with license info
gitleaks	8.30.1	✅ No leaks	163 commits, 94 MB scanned
scorecard	v5.5.0	✅ 5.0/10	See breakdown below
syft	1.44.0	✅ Generated	43 components (GitHub Actions only)
compliance-assistant (enrich)	—	⚠️ Failed	GitHub API 404
grype	0.112.0	✅ 2 findings	Both in GitHub Actions, not in project code

Scorecard breakdown:

Check	Score	Notes
Binary-Artifacts	10/10	
Branch-Protection	0/10	Not enabled
CI-Tests	10/10	
CII-Best-Practices	0/10	No badge
Code-Review	0/10	Single developer
Contributors	6/10	2 orgs (Matdata + Copilot)
Dangerous-Workflow	10/10	
Dependency-Update-Tool	0/10	Not configured
Fuzzing	0/10	
License	10/10	
Maintained	10/10	Active
Packaging	10/10	
Pinned-Dependencies	0/10	Actions not pinned by hash
SAST	0/10	
Security-Policy	10/10	
Signed-Releases	−1/10	No releases
Token-Permissions	0/10	Excessive permissions
Vulnerabilities	10/10	